KRR and Epistemic Planning: Specification and Implementation Issues

Tran Cao Son

Department of Computer Science
New Mexico State University
MSC CS, PO Box 30001
Las Cruces, New Mexico 88003

Dagstuhl 2017

Outline

Outline

Epistemic planning in single agent environment

Need for reasoning about knowledge (or beliefs) of agents in planning

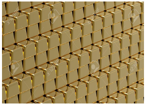
Example: Open the correct door and you get the gold; the wrong one and meet a tiger!

Epistemic planning in single agent environment

Need for reasoning about knowledge (or beliefs) of agents in planning

Example: Open the correct door and you get the gold; the wrong one and meet a tiger!

Real state of the world



Epistemic planning in single agent environment

Need for reasoning about knowledge (or beliefs) of agents in planning

Example: Open the correct door and you get the gold; the wrong one and meet a tiger!

What is a plan? Open a door (left or right)? This does not guarantee success.

Epistemic planning in single agent environment

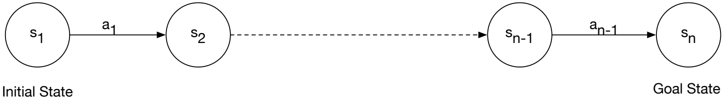
Need for reasoning about knowledge (or beliefs) of agents in planning

Example: Open the correct door and you get the gold; the wrong one and meet a tiger!

What is a plan? Open a door (left or right)? This does not guarantee success.
A reasonable plan: determine where the tiger is (e.g., smell, or make noise then listen, etc.) and open the other door.

Rough classification

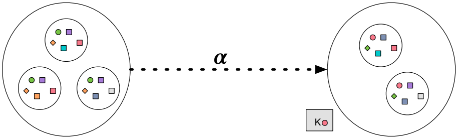
- Conformant planning: initial state is incomplete, no sensing action, actions might be non-deterministic; solution is a sequence of actions (s_i is a belief state, a_i is an action).



- Conditional planning: initial state is incomplete, sensing action, actions might be non-deterministic (probabilistic); plan is often a policy or a conditional plan (with **if-then** constructs).

State of the art

- Several approaches to planning with incomplete information and sensing actions in single agent environment.
- Available systems: generation of *plan* satisfying
  $(Domain, Initial\_State) \models \mathbf{K}\,\varphi$ **after** *plan*



- No system can be used to determine whether the following holds for every *plan* (of bounded length)

  $$(Domain, Initial\_State) \not\models \neg(\mathbf{K}\,\varphi \vee \mathbf{K}\,\neg\varphi)\ \textbf{after}\ \ plan$$

  - practical application: security.
  - strong tie to multi-agent system.

Outline

1. Languages for representation of dynamic domains (or reasoning about actions and their effects).
2. Basic algorithms for computing successor states.
3. Search algorithms for plan generation.

Important Notions

1. State
2. Plan

This presentation

Focus on domains with binary fluents, no probabilistic actions.

Outline

Example

- *Problem*: John is at home and his car is at home. He needs to board the plane to go to Dagstuhl.
- *Question*: What should John do?
- *Solution*: Drive to the airport. Look for the gate. Go to the gate. Board the plane.

Language for RAC or Planning

- situation calculus [McCarthy and Hayes (1969)]
- action language [Gelfond and Lifschitz (1993)]
- fluent calculus [Thielscher (2000)]
- PDDL [Ghallab et al. (1998)]

Most have been extended to dealing with sensing actions and incomplete information.

Basic Ontologies (Situation Calculus, [McCarthy and Hayes (1969)])

- Situation: a complete state of the universe in an instance of time, often given by a set of facts
  - The fact "John is at home" is represented by the atom $at(john, home)$.
  - "His car is at home" is another fact, that can be represented by the atom $at(car, home)$.
- Fluent: a function whose domain is the space of situations
  E.g. $at(john, home)$ is a Boolean function whose domain is the set of situations, $at(john, home)(s)$ is true says that "John is at home in situation s."
- Action: causes for changes from situations to situations
  E.g. $drive(home, airport)$ is an action that changes the situation in which John is at home to the situation in which John is at the airport.

Basic Ontologies (Situation Calculus, [Reiter (2001)])

- Situation: a possible history of the world
  - s_0 – initial situation.
  - $do(drive(home, airport), s_0)$ – situation after the execution of $drive(home, airport)$ in s_0.
- Fluent: a relation (a property of the world) whose (truth) value changes over time due to the execution of actions
  - $at(john, home)$ is a relation whose truth value changes – a *Boolean* fluent.
  - $number\_paper(john)$ is a relation whose value changes – a *functional* fluent.
- Action: causes for *all* changes in the world
  E.g. $drive(home, airport)$ is the *only action* that can change the world in our example.

Basic Ontologies (Action Languages, [Gelfond and Lifschitz (1993)])

- Actions and fluents – same as in situation calculus in [Reiter (2001)]
- Fluent literal – a fluent or its negation (a fluent preceeding by $\neg$)
  E.g. $at(john, home)$, $\neg at(john, home)$
- State: two commonly used definitions
  - a set of fluents or
  - a *complete* and *consistent* set of fluent literals, i.e., s is a state if for every fluent f
    - either f or $\neg f$ belongs to s; and
    - $\{f, \neg f\} \not\subseteq s$.

We will use the ontologies of action languages in this presentation.

Action Language $\mathcal{AL}$ — Syntax

- Fluents and Actions: disjoint sets of propositional symbols (e.g., fluent: $at(john, home)$, action: $drive(home, airport)$)
- Laws:
  - *Dynamic law*: describes effects of world altering actions

    $$drive(home, airport) \textbf{ causes } at(john, airport), at(car, airport)$$

  - *Knowledge law*: describes effects of sensing actions

    $$look \textbf{ determines } at(plane, gate(1))$$

  - *Static causal law*: represents the relationship between fluents

    $$\neg at(john, home) \textbf{ if } at(john, airport)$$

  - *Executability law*: encodes the conditions under which an action can be executed

    $$drive(home, airport) \textbf{ executable } at(john, home), at(car, home)$$

  - *Initial state*: a set of fluent literals

Action Theory — Syntax

Definition

An action theory is a pair (D, δ) where

- D, called an *action domain*, is a set of dynamic, knowledge, static causal, and executability laws.
- δ, called the *initial state*, is a set of fluent literals.

(D_a, δ_a)—"Going to Dagstuhl" Action Theory

$$D_a = \begin{cases} drive(home, airport) \textbf{ executable } at(john, home), at(car, home) \\ drive(home, airport) \textbf{ causes } at(john, airport), at(car, airport) \\ board(gate(1)) \textbf{ causes } in\_plane \textbf{ if } at(plane, gate(1)), at(john, gate(1)) \\ \ldots \\ look \textbf{ determines } at(plane, gate(1)) \\ \neg at(john, airport) \textbf{ if } at(john, home) \quad \neg in\_plane \textbf{ if } at(john, home) \\ \ldots \end{cases}$$

$$\delta_a = \{at(john, home), at(car, home), \neg in\_plane, \neg at(john, airport), \neg at(car, airport)\}$$

Action language $\mathcal{AL}$ (Semantics) — Intuition

Given an action theory (D, δ), the action domain D encodes a transition system consisting of elements of the form $\langle s_1, a, s_2 \rangle$ where s_1 and s_2 are states of the theory and a is an action that, when executed in s_1, changes the state of the world from s_1 into s_2. For example, in (D_a, δ_a)

$$D_a = \begin{cases} drive(home, airport) \textbf{ executable } \ at(john, home), at(car, home) \\ drive(home, airport) \textbf{ causes } \ at(john, airport), at(car, airport) \\ board(gate(1)) \textbf{ causes } \ in\_plane \textbf{ if } \ at(plane, gate(1)), at(john, gate(1)) \\ \ldots \\ look \textbf{ determines } at(plane, gate(1)) \\ \neg at(john, airport) \textbf{ if } \ at(john, home) \quad \neg in\_plane \textbf{ if } \ at(john, home) \\ \ldots \end{cases}$$

$\delta_a = \{at(john, home), at(car, home), \neg in\_plane, \neg at(john, airport), \neg at(car, airport)\}$
a transition is

$$\begin{gathered}\langle\{at(john, home), at(car, home), \neg in\_plane, \neg at(john, airport), \neg at(car, airport)\}, \\ drive(home, airport), \\ \{\neg at(john, home), \neg at(car, home), \neg in\_plane, at(john, airport), at(car, airport)\}\rangle\end{gathered}$$

Action language $\mathcal{AL}$, Complete Information (Semantics)

Given an action domain D, a fluent literal l, sets of fluent literals σ and ψ

- $\sigma \models l$ iff $l \in \sigma$; $\sigma \models \psi$ iff $\sigma \models l$ for every $l \in \psi$.
- σ satisfies a static causal law φ **if** ψ if $\sigma \models \psi$ implies that $\sigma \models \varphi$.
- Closure: $Cn_D(\sigma)$, called the closure of σ, is the smallest set of literals that contains σ and satisfies all static causal laws
- State: *complete* and *consistent* set of fluent literals which *satisfies all* static causal laws.
- Transition Function:
  $\Phi : Actions \times States \to States$ where

$$\Phi(a, s) = \begin{cases} \{s' \mid s' = Cn_D(de(a, s) \cup (s \cap s'))\} \\ \text{ if } D \text{ contains } a \textbf{ executable } \varphi \text{ and } s \models \varphi \\ \\ \Phi(a, s) = \emptyset \quad \text{otherwise} \end{cases}$$

Outline

Approaches

- Possible world approach (PSW): Extension of the transition function Φ to a transition function Φ^c over belief states.
- Approximation: Modifying the transition function Φ to define a transition function Φ^a over approximation states.

Notation

	Belief states (S and Σ)	Approximation states (δ and Δ)	
S	a set of states	a set of fluent literals	δ
Σ	a set of belief states	a set of approximation states	Δ

Semantics (for deterministic case)

- PSW: from $\Phi : Action \times States \rightarrow 2^{States}$ to $\Phi^c : Action \times 2^{States} \rightarrow 2^{2^{States}}$

$$\Phi^c(a, S) = \begin{cases} \{\Phi(a, s) \mid s \in S\} & a \text{ is a world altering action,} \\ & \text{executable in } S \\ \{S_f, S_{\neg f}\} & a \text{ is a sensing action, executable in } S \\ & a \textbf{ determines } f \in D \end{cases}$$

  where $S_f = \{s \mid s \in S, s \models f\}$, $S_{\neg f} = \{s \mid s \in S, s \models \neg f\}$

- Approximation: several approximations exist, e.g., 0-approximation $\Phi^0 : Action \times Partial\_States \rightarrow Partial\_States$

$$\Phi^0(a, \delta) = (\delta \cup de(a, \delta)) \setminus \neg pe(a, \delta)$$

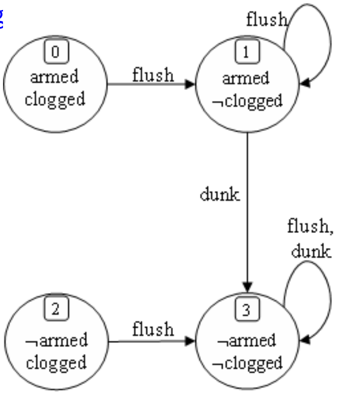
  where
  - $de(a, \delta)$ is the set of "direct effects" of a in δ
  - $pe(a, \delta)$ is the set of "possible effects" of a in δ

Example (Bomb-In-The-Toilet)

There may be a bomb in a package. Dunking the package into a toilet disarms the bomb.

- Fluents: *armed*, *clogged*
- Actions: *dunk*, *flush*
- Action domain:

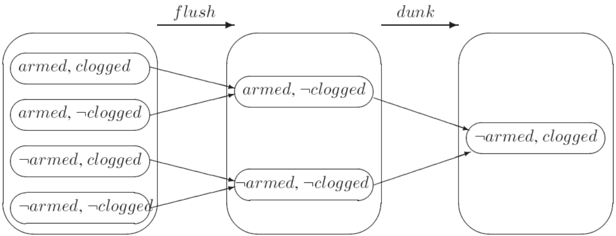
$$\mathcal{D}_b = \begin{cases} \textit{dunk}\ \textbf{causes}\ \neg \textit{armed}\ \textbf{if}\ \textit{armed} \\ \textit{flush}\ \textbf{causes}\ \neg \textit{clogged} \\ \textit{dunk}\ \textbf{executable}\ \neg \textit{clogged} \end{cases}$$



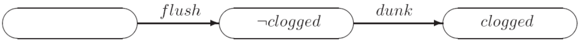
- Initially, we know nothing about the value of *armed* and *clogged*.
- PWS: the initial belief state $S_0 = \{0, 1, 2, 3\}$.
- Approximation: the initial approximation state $\delta_0 = \emptyset$.

Illustration

PSW-Approach



0-Approximation



Outline

1. Epistemic Planning Problem
   - Motivation
   - Research Issues
2. Single Agent Environment
   - Specification
   - Reasoning about Sensing Actions and Incomplete Information
   - **Planning with Incomplete Information and Sensing Actions**
   - Summary
3. Multi Agent Environment
   - Background
   - DEL Formalization
   - An Approximation of DEL for Epistemic Planning Specification: Action Language $mA+$
   - Specifying Initial State
4. Summary and Discussion

Complexity

Definition (Planning Problem)

- Given: an $\mathcal{AL}$-action theory (D, δ), where δ is a partial state, and a set of fluent literals G.
- Determine: a sequence of actions α such that $(D, \delta) \models G$ **after** α

From [Baral et al. (2000); Liberatore (1997); Turner (2002)]:

Theorem (Complexity)

- Planning*: (D, δ) is deterministic: Σ_P^2-hard even for plans of length 1, Σ_P^2-complete for polynomial-bounded length plans.*
- Planning*: (D, δ) is non-deterministic: Σ_P^3-hard even for plans of length 1, Σ_P^3-complete for polynomial-bounded length plans.*

Planning Algorithms

(1) *Heuristic search based approaches*

- State space: the search space is the set of possible states (belief states/partial states)
- Plan space (partial order planning): the search space is the set of possible plans

(2) *Translation based approaches* (SAT-, model checking-, or answer set solvers).

- SAT: translation into a SAT instance
- Model checking: translation into a model checking problem
- Answer set programming: translation into a logic program

Approximation based planning

- Address the complexity problem of the possible world approach: give up completeness for efficiency in reasoning/planning
- Sound with respect to possible world semantics (formal proof is provided in some work)
- Representation languages and approaches are different
  - Situation calculus: [Etzioni et al. (1996); Goldman and Boddy (1994); Petrick and Bacchus (2004)]
  - Action languages: [Son and Baral (2001); Son and Tu (2006); Son et al. (2005b)].
  - Logic programming: [Son et al. (2005a); Tu et al. (2011)].

What is good about the approximation?

Theorem (Complexity)

Conformant Planning: *(D, δ) is deterministic: NP-complete for polynomial-bounded length plans.*

Consequence

If (D, δ) is complete (satisfying certain conditions), planners can use the 0-approximation (lower complexity) instead of the possible world semantics. In fact, classical planners can be used to solve conformant planning (change in the computation of the next state.)

Approximation Based Conformant Planners

- Earlier systems [Etzioni et al. (1996); Goldman and Boddy (1994)]: approximation is used in dealing with sensing actions (context-dependent actions and non-deterministic outcomes)
- PKS [Petrick and Bacchus (2004)] is very efficient (plus: use of domain knowledge in finding plans)
- CpA and several of its improvements [Tu et al. (2006, 2011); Tran et al. (2013); To et al. (2015)] are competitive with others such as CFF, POND, and KACMBP in several benchmarks
- Completeness can be achieved via reasoning algorithms that simulate the possible world model approach in the space of incomplete states.

Outline

1. Study in reasoning about actions and changes might provide useful ways for dealing with complex domains
2. Approximations can compensate for the inaccuracy of heuristics
3. Approximations can be useful when the computation of the next state is more complicated
4. Completeness conditions can be used to deal with sensing actions in conditional planners: deciding when to execute a sensing action?

Outline

Single-agent planning

- deliberation process for generating a plan that transforms the state of the world from an initial state to a state satisfying a predefined goal

Multi-agent planning

- generalization of the single-agent planning problem to domains where several agents plan and act together and have to share resources, activities, and goals

(From "Automated Planning–Theory and Practice" by Ghallab, Nau, & Traverso.)

Single-agent planning

- Specification of the capabilities of agents, initial state, goal (e.g., (D, I, G));
- The deliberation process is supported by an entailment relation $(D, I) \models G$ **after** α;
- Different types of planning (complete vs. incomplete, optimal, etc.);
- Benchmarks, several planning systems available.

Multi-agent planning

- Specification of the capabilities of single agents is not enough (interactions are important);
- What is the generalization of the entailment relation $\models$? (what is needed?)
- Earlier literature: focus on coordinating of agent activities, resources, and goals;
- Benchmarks, systems hard to come by.

Multi-agent planning: a bit of history

- Earlier approaches: planning for multiple agents; do not consider knowledge, beliefs, or privacy of agents, e.g., [Allen and Zilberstein (2009); Bernstein et al. (2002); Durfee (1999); Guestrin et al. (2001); Nair et al. (2003); Peshkin and Savova (2002); de Weerdt and Clement (2009)].
- Addressing the privacy concern: MA-STRIPS; decentralized planning; communication via public fluents, e.g., [Brafman and Domshlak (2008); Brenner (2003); Nissim and Brafman (2012); Torreño et al. (2012)].
- Generalization of POMPD: decentralized-POMPD (fully cooperative agents) [de Weerdt and Clement (2009); Shoham and Leyton-Brown (2011)]; I-POMPD (self-interested agents) [Rathnasabapathy et al. (2006); Poupart and Boutilier (2003); Sonu and Doshi (2014)]
- Epistemic planning: considering manipulation of knowledge and beliefs of agents; (no) common knowledge [Crosby et al. (2014); Muise et al. (2015); Wan et al. (2015)]

Outline

Epistemic language and model

$\mathcal{A}$: set of agents; P: set of propositions.

Multi-agent epistemic logic language $L(P, \mathcal{A})$

$$\varphi \stackrel{def}{=} \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid C_X\varphi$$

where $p \in P$, $i \in \mathcal{A}$, $X \subseteq \mathcal{A}$.
$K_i\varphi$: "agent i knows φ" and
$C_X\varphi$: "the agents in X share the knowledge about φ"

Epistemic model

$\mathcal{M} = (W, R, \pi)$, where *(i)* W is the domain, a finite set of worlds; *(ii)* $R : \mathcal{A} \to 2^{W \times W}$ assigns an accessibility relation R_i to each agent $i \in \mathcal{A}$. *(iii)* $\pi : P \to 2^W$: valuation of that variable.
A pointed epistemic model is a pair $(\mathcal{M}, w)$ where $\mathcal{M} = (W, R, \pi)$ and $w \in W$.

Satisfaction of formulas w.r.t. pointed epistemic models

Given: $(\mathcal{M}, w)$ with $\mathcal{M} = (M, R, \pi)$ and a formula φ, $(\mathcal{M}, w) \models \varphi$ is defined as follows:

- $(\mathcal{M}, w) \models \top$ always;
- $(\mathcal{M}, w) \models \bot$ never;
- $(\mathcal{M}, w) \models p$ iff $w \in \pi(p)$;
- $(\mathcal{M}, w) \models \neg\varphi$ iff $(\mathcal{M}, w) \not\models \varphi$;
- $(\mathcal{M}, w) \models \varphi_1 \wedge \varphi_2$ iff $(\mathcal{M}, w) \models \varphi_1$; and $(\mathcal{M}, w) \models \varphi_2$;
- $(\mathcal{M}, w) \models K_i\varphi$ if for all $v \in W$, if wR_iv then $(\mathcal{M}, v) \models \varphi$; and
- $(\mathcal{M}, w) \models C_X\varphi$ if for all $v \in W$, if $w(\bigcup_{j\in X} R_j)^* v$ then $(\mathcal{M}, v) \models \varphi$

where $(\bigcup_{j\in X} R_j)^*$ is the transitive closure of $\bigcup_{j\in X} R_j$.

$\boxed{\mathcal{M} \models \varphi \text{ if } (\mathcal{M}, w) \models \varphi \text{ for each } w \in W.}$

Axioms, Knowledge and Beliefs

$\mathcal{M} = (S, R, \pi)$: an epistemic model

- **K** $\stackrel{def}{=} \forall i \in \mathcal{A}, \varphi, \psi \in L(P, \mathcal{A}).[\mathcal{M} \models (K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi]$;
- **T** $\stackrel{def}{=} \forall i \in \mathcal{A}, \psi \in L(P, \mathcal{A}).[\mathcal{M} \models K_i\psi \Rightarrow \psi]$;
- **4** $\stackrel{def}{=} \forall i \in \mathcal{A}, \psi \in L(P, \mathcal{A}).[\mathcal{M} \models K_i\psi \Rightarrow K_iK_i\psi]$;
- **5** $\stackrel{def}{=} \forall i \in \mathcal{A}, \psi \in L(P, \mathcal{A}).[\mathcal{M} \models \neg K_i\psi \Rightarrow K_i\neg K_i\psi]$; and
- **D** $\stackrel{def}{=} \forall i \in \mathcal{A}, \psi \in L(P, \mathcal{A}).[\mathcal{M} \models \neg K_i \perp]$.

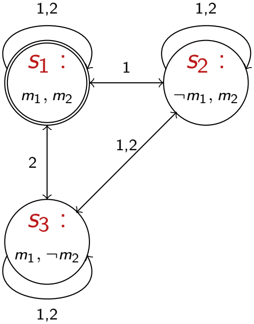
- $\mathcal{M}$ is **T**- (**4**-, **K**-, **5**-, **D**-, respectively) model if it satisfies property **T** (**4**, **K**, **5**, **D**, respectively).
- $\mathcal{M}$ is a **S5** model if it satisfies the properties **K**, **T**, **4**, and **5**.
- $\mathcal{M}$ is a **KD45** model if it satisfies the properties **K**, **D**, **4**, and **5**.

Epistemic state

An epistemic state is a pair $(\mathcal{M}, W_d)$ where $\mathcal{M} = (M, R, \pi)$ is an epistemic model and $W_D \subseteq W$. A truth value of a formula φ with respect to an epistemic state $(\mathcal{M}, W_d)$ is defined by

$$(\mathcal{M}, W_d) \models \varphi \qquad \text{iff} \qquad \forall w \in W_d.[(\mathcal{M}, w) \models \varphi]$$

An epistemic state of the muddy children example with two kids

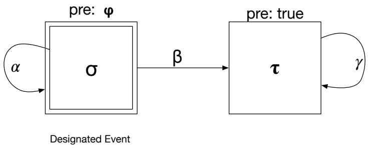


Event model

An event model for $L(P, \mathcal{A})$ is a quadruple $E = (E, Q, pre, post)$ where:

- E is a finite non-empty set of events;
- $Q : \mathcal{A} \to 2^{E \times E}$ assigns an accessibility relation to each agent $i \in \mathcal{A}$; $pre : E \to L(P, \mathcal{A})$ assigns to each event a precondition; and
- $post : E \to L(P, \mathcal{A})$ assigns to each event a postcondition.

A pair (E, E_d) consisting of an event model $E = (E, Q, pre, post)$ and a non-empty set of designated events $E_d \subseteq E$ is called an epistemic action.



Graphical Representation of Epistemic Action

Action execution

Given an epistemic action (E, E_d) and an epistemic state $(\mathcal{M}, W_d)$

- (E, E_d) is executable in $(\mathcal{M}, W_d)$ if for each $w \in W_d$ there exists at least one $e \in E_d$ such that $(\mathcal{M}, w) \models pre(e)$.
- $(\mathcal{M}, W_d) \otimes (E, E_d) = ((W', R', \pi'), W'_d)$ where
    - $W' = \{(w, e) \in W \times E \mid (\mathcal{M}, w) \models pre(e)\}$
    - $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$
    - $\pi'(p) = (\{(w, e) \in W' \mid (\mathcal{M}, w) \models p\} \setminus \{(w, e) \in W' \mid post(e) \models \neg p\}) \cup \{(w, e) \in W' \mid post(e) \models p\}$
    - $W'_d = \{(w, e) \in W' \mid w \in W_d \text{ and } e \in E_d\}$

  if (E, E_d) is *executable* in $(\mathcal{M}, W_d)$.

$\otimes$ plays the role of the function Φ in single agent domains.

Epistemic planning domain

An epistemic planning domain on $(P, \mathcal{A})$ is a restricted state-transition system $\Sigma = (S, A, \gamma)$ where

- S is a finite or recursively enumerable set of epistemic states of $L(P, \mathcal{A})$,
- A is a finite set of epistemic actions of $L(P, \mathcal{A})$, and
- γ is defined by

$$\gamma(s, a) = \begin{cases} s \otimes a & \text{if } a \text{ is executable in } s \\ \textit{undefined} & \text{otherwise} \end{cases}$$

Epistemic planning problem and solution

An epistemic planning problem is a triple (Σ, s_0, ϕ_g) where $\Sigma = (S, A, \gamma)$ is an epistemic planning domain on $(P, \mathcal{A})$, $s_0 \in S$ is the initial state, and ϕ_g is a formula in $L(P, \mathcal{A})$.

(Σ, s_0, ϕ_g): an epistemic planning problem and $S_g = \{s \in S \mid s \models \phi_g\}$. An action sequence $a_1, \ldots, a_n$ such that

$$\gamma(\gamma(\ldots\gamma(\gamma(s_0, a_1), a_2), \ldots, a_{n-1}), a_n) \in S_g$$

(or, $s_0 \otimes a_1 \otimes a_2 \otimes \ldots \otimes a_n) \models \phi_g$) is a solution of the problem.

Advantages

- Strongly connected to the research in Dynamic Epistemic Logic (DEL) (e.g., [Baltag and Moss (2004); Boella and van der Torre (2005); Fagin et al. (1995); Gerbrandy (2006); Herzig and Troquard (2006); Meyer (2000); Sauro et al. (2006); Spaan et al. (2006); van Benthem et al. (2006); van der Hoek et al. (2005); van Ditmarsch et al. (2007))])
- Useful for formalizing different types of epistemic planning (different perspectives), e.g.,
  - $(\mathcal{M}, W_d)$ with $|W_d| = 1$: external observer;
  - $(\mathcal{M}, W_d)$ with W_d is closed under R_i: view of agent i.
- Useful for complexity study (e.g., [Aucher and Bolander (2013); Bolander et al. (2015)]).

Difficulties in development of epistemic planner

Given (Σ, s_0, ϕ_g) where $\Sigma = (S, A, \gamma)$:

- Specification
  - How do we specify an epistemic planning problem? (Σ)
  - How do we specify the set of epistemic actions? (A) (art vs. craft)
  - The set of states S consists of only **S5**-models.
  - Do we need to distinguish knowledge from beliefs?
- Implementation: data structure, heuristic

PDDL like specification language for epistemic planning?

Difficulties in development of epistemic planner

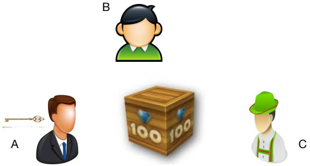
Given (Σ, s_0, ϕ_g) where $\Sigma = (S, A, \gamma)$:

- Specification
  - How do we specify an epistemic planning problem? (Σ)
    - the set of epistemic models over the language $L(P, \mathcal{A})$ is normally infinite.
    - ensuring that $s \otimes a \in S$, for each $s \in S$ and each $a \in A$ is not easy.
  - How do we specify the set of epistemic actions? (A) (art vs. craft)
    - What is the epistemic action of "i is an agent among a group of agents and some agent makes the proposition b false while agent i is not present"?
  - The set of states S consists of only **S5**-models.
  - Do we need to distinguish knowledge from beliefs?
- Implementation: data structure, heuristic

PDDL like specification language for epistemic planning?

Outline

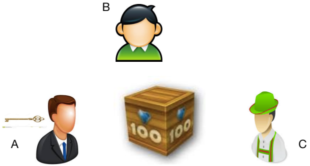
Example: Three Agents and the Coin Box



- Nobody knows whether the coin lies heads or tails up;
- The box is locked; needs key to open; only A has the key of the box;
- Peeking into an open box will learn whether the coin lies heads or tails up;
- Observing someone peeking into the box will allow an agent to know that the other agent knows the status of the coin but does not allow one knows the status of the coin if one does not know it already;
- Distracting an agent causes that agent to not look at the box;
- Signaling an agent to look at the box causes this agent to look at the box;
- Announcing that the coin lies heads or tails up will make this a common knowledge among the agents that are listening.

Example: Three Agents and the Coin Box

- Nobody knows whether the coin lies heads or tails up;
- The box is locked; needs key to open; only A has the key of the box;

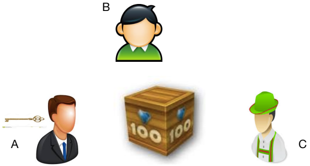


- Peeking into an open box will learn whether the coin lies heads or tails up;
- Observing someone peeking into the box will allow an agent to know that the other agent knows the status of the coin but does not allow one knows the status of the coin if one does not know it already;
- Distracting an agent causes that agent to not look at the box;
- Signaling an agent to look at the box causes this agent to look at the box;

Can A know the status of the coin, let B know that she knows it, and does not allow C to be aware of it?

Example: Three Agents and the Coin Box

- Nobody knows whether the coin lies heads or tails up;
- The box is locked; needs key to open; only A has the key of the box;



- Peeking into an open box will learn whether the coin lies heads or tails up;
- Observing someone peeking into the box will allow an agent to know that the other agent knows the status of the coin but does not allow one knows

Can A know the status of the coin, let B know that she knows it, and does not allow C to be aware of it?

- Distracting C from looking at the box;
- Signaling B to look at the box;
- Opening the box; and
- Peeking into the box.

Language $mA+$: Syntax

Different types of actions:

- World altering action: opening a box
- Sensing action: peeking into the box

Special for multi-agent environment

- Announcement action: announcing that the coin lies head (or tail) up
- Manipulating observability: distracting another agents from watching self (or signaling another agents to watch self)
- Manipulating beliefs: peeking while other agents are looking

Language $mA+$: Syntax—Specification of actions and effects

World altering action: opening a box

$open(X)$ **causes** $opened$ and
$open(X)$ **executable** $has\_key(X)$

Sensing action: peeking into the box

$peek(X)$ **determines** $tail$ and
$peek(X)$ **executable** $opened, looking(X)$

Announcement action: announcing that the coin lies head (or tail) up

$shout\_tail(X)$ **announces** $tail$

Language $mA+$: Syntax—Specification of actions and effects

World altering action: opening a box

$open(X)$ **causes** $opened$ and
$open(X)$ **executable** $has\_key(X)$

How about?

- Manipulating observability: distracting another agents from watching self (or signaling another agents to watch self)
- Manipulating beliefs: peeking while other agents are looking

Announcement action: announcing that the coin lies head (or tail) up

$shout\_tail(X)$ **announces** $tail$

Language $mA+$: Syntax—Specification of actions and effects

World altering action: opening a box

$open(X)$ **causes** $opened$ and
$open(X)$ **executable** $has\_key(X)$

How about?

- Manipulating observability: distracting another agents from watching self (or signaling another agents to watch self)
  this is world altering action!
- Manipulating beliefs: peeking while other agents are looking
  this is sensing action!

Need: specification of observability

Language *mA+*: Syntax—Specification of observability

Classification of observability

- Full observers: those who observe the action occurrence and fully aware of its effects
- Partial observers: those who observe the action occurrence but do not know of its effects
- Oblivious: those who are not aware of the action occurrence.

Possible classification of observability

action type	full observers	partial observers	oblivious
world-altering actions	$\surd$		$\surd$
sensing actions	$\surd$	$\surd$	$\surd$
announcement actions	$\surd$	$\surd$	$\surd$

Language $mA+$: Syntax—Specification of observability

Three Agents and Coin Box

$X, Y \in \{A, B, C\}, X \neq Y$:

X **observes** $open(X)$	X full observer
X **observes** $peek(X)$	$--$
Y **observes** $open(X)$ **if** $looking(Y)$	$--$
Y **aware_of** $peek(X)$ **if** $looking(Y)$	Y partially observer
Y **observes** $shout\_tail(X)$	X full observer
$\{X, Y\}$ **observes** $distract(X, Y)$	X, Y full observer
$\{X, Y\}$ **observes** $signal(X, Y)$	$--$

Language $mA+$: Semantics

Domain

A set of statements about action effects and observability over the pair $(P, \mathcal{A})$ is a multi-agent domain.

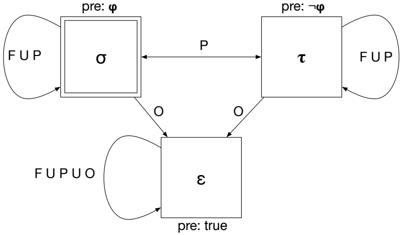
A multi-agent domain specifies a collection of epistemic actions defined as follows. Given an epistemic model $(\mathcal{M}, W_d)$ and an action occurrence a,

Frame of reference $\rho = (F, P, O)$

- F - the set of agents who are fully observer of the occurrence.
- P - the set of agents who are partially observer of the occurrence.
- O - the set of agents who are oblivious of the occurrence.

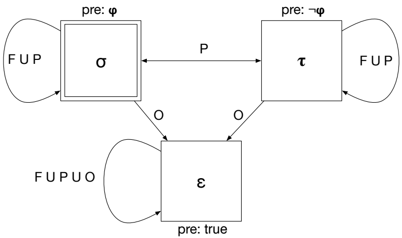
Announcement

a announces φ

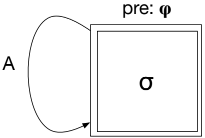


Announcement

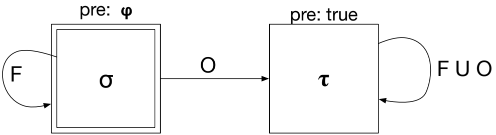
a announces φ



Special Case: $(F \cup O = \emptyset$ and $P = \emptyset)$



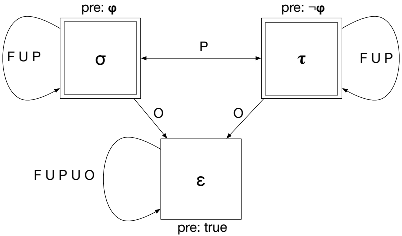
Public Announcement



Private Announcement

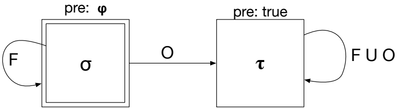
Sensing Action

a determines φ



World Altering Action

a **executable** φ



- $pre(\sigma) = \varphi$ and $pre(\tau) = \top$; and
- $sub(\tau) = \emptyset$ and $sub(\sigma) = \{p \rightarrow \Psi^+(p, a) \vee (p \wedge \neg\Psi^-(p, a)) \mid p \in P\}$, where
  $\Psi^+(p, a) = \bigvee\{\varphi \mid [a \textbf{ causes } p \textbf{ if } \varphi] \in D\}$ and
  $\Psi^-(p, a) = \bigvee\{\varphi \mid [a \textbf{ causes } \neg p \textbf{ if } \varphi] \in D\}$.

Advantages of $mA+$

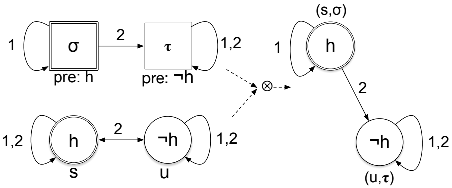
Under certain conditions, $\mathcal{M} \otimes E$ maintains **KD45** property of $\mathcal{M}$

Advantages of $mA+$

Under certain conditions, $\mathcal{M} \otimes E$ maintains **KD45** property of $\mathcal{M}$

Why **KD45**?

- $\otimes$ does not maintain **S5** if private action is considered.



- Distinguishing between knowledge and belief is necessary.
- In **KD45**, knowledge is true belief and so only one modal operator (belief) is necessary [Halpern et al. (2009)].

Advantages of $mA+$

Under certain conditions, $\mathcal{M} \otimes E$ maintains **KD45** property of $\mathcal{M}$

Why **KD45**?

- $\otimes$ does not maintain **S5** if private action is considered.
- Distinguishing between knowledge and belief is necessary.
- In **KD45**, knowledge is true belief and so only one modal operator (belief) is necessary [Halpern et al. (2009)].

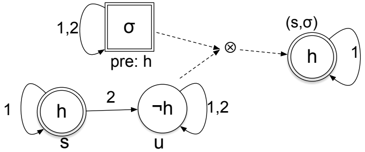
Verifying **KD45**

$\mathcal{M} = (S, R, \pi)$: **KD45** model if it satisfies **K**, **D**, **4**, and **5**.
$\mathcal{M}$ is **KD45** iff every R_i in R is serializable, transitive, and Euclidean.

- serial iff for every $u \in S$ there exists some $v \in S$ s.t. $(u, v) \in R_i$;
- transitive iff $(u, v) \in R_i \wedge (v, z) \in R \Rightarrow (u, z) \in R_i$;
- Euclidean iff $(u, v) \in R_i \wedge (u, z) \in R \Rightarrow (v, z) \in R_i$.

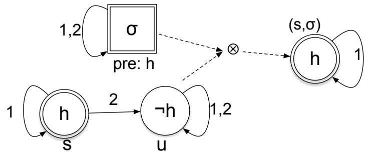
KD45 is not maintainable

Loss of seriality

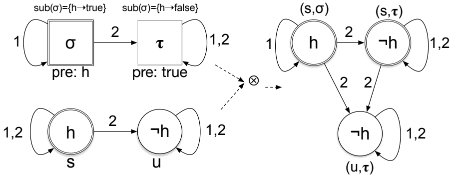


KD45 is not maintainable

Loss of seriality



Loss of semi-reflexivity — leading to loss of seriality



Conditions for **KD45** maintainability

Semi-reflexive Kripke Model

$\mathcal{M}$ is *semi-reflexive* if for every agent i and state u in $\mathcal{M}$, there exists some v such that (u, v) belongs to the accessibility relation of i and u and v have the same representation (no false belief!).

Primitive Action Model

A serial, transitive, and Euclidean update model is *primitive* if the precondition $pre(\sigma)$ at every event σ is an atomic formula and for every agent i and $(\sigma, \tau) \in R_i$ such that $\sigma \neq \tau$ then either *(i)* $(\sigma, \sigma) \in R_i$; or *(ii)* $pre(\tau) = true$ and $sub(\tau) = sub(\sigma) = \emptyset$. (no secret changing the world!)

- $\mathcal{M} \otimes \mathbf{\Sigma}$ is semi-reflexive if $\mathcal{M}$ is semi-reflexive and $\mathbf{\Sigma}$ is primitive, serial, transitive, and Euclidean action model.
- Several theories in $mA+$ **can be represented using primitive action models**.

Outline

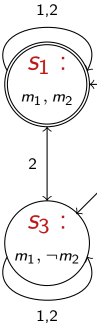
Literature

- Given a set of formulas Γ in the language $L(P, \mathcal{A})$.
- Identify an epistemic model satisfying Γ.

Literature

- Given a set of formulas Γ in the language $L(P, \mathcal{A})$.
- Identify an epistemic model satisfying Γ.

The muddy children example - the initial state is as follows



Literature

- Given a set of formulas Γ in the language $L(P, \mathcal{A})$.
- Identify an epistemic model satisfying Γ.

Question?

- Is the identified model unique?
- How to compute the identified model?

Literature

- Given a set of formulas Γ in the language $L(P, \mathcal{A})$.
- Identify an epistemic model satisfying Γ.

Well-known properties

Multi-agent domains:

- Models of Γ can be infinite.
- If Γ is consistent (has a model) then it has a finite model.
- Adding common knowledge operator C usually increases complexity.
- In multi-modal logics, a theory can have infinitely many infinite models even for finite set of propositions.

Solution

Identifying **S5**-theories which have finitely many finite models

Requirement

sufficiently expressive to represent problems with common knowledge found in the literature.

Observations

- all agents share some knowledge (e.g., 1 and 2 know that at least one of them is muddy);
- common knowledge among group of agents about the state of knowledge of some agent with respect to a certain property (e.g., 1 knows that 2 does not know whether or not he is muddy)

A Proposal: Primitive Finitary **S5**-Theory

Complete clause

φ: complete clause if $\varphi = \bigvee_{p \in P} p^*$ where p^* is either p or $\neg p$.

Primitive Finitary **S5**-Theory

- Each formula in T is of the form
  1. φ
  2. $\mathbf{C}(\mathbf{B}_i\varphi)$ or
  3. $\mathbf{C}(\mathbf{B}_i\varphi \vee \mathbf{B}_i\neg\varphi)$ or
  4. $\mathbf{C}(\neg\mathbf{B}_i\varphi \wedge \neg\mathbf{B}_i\neg\varphi)$.
- For each complete clause φ over P and each agent i, T contains (2), (3), or (4)

A Proposal: Primitive Finitary **S5**-Theory

Complete clause

φ: complete clause if $\varphi = \bigvee_{p \in P} p^*$ where p^* is either p or $\neg p$.

Primitive Finitary **S5**-Theory

- Each formula in T is of the form

Intuition

1. φ: properties that are true in the actual world;
2. $\mathbf{C}(\mathbf{B}_i \varphi)$: common knowledge: i knows the truth value of φ is true;
3. $\mathbf{C}(\mathbf{B}_i \varphi \vee \mathbf{B}_i \neg\varphi)$: common knowledge: i knows the truth value of φ;
4. $\mathbf{C}(\neg\mathbf{B}_i \varphi \wedge \neg\mathbf{B}_i \neg\varphi)$: common knowledge: i does not know the truth value of φ.

A Proposal: Primitive Finitary **S5**-Theory

Complete clause

φ: complete clause if $\varphi = \bigvee_{p \in P} p^*$ where p^* is either p or $\neg p$.

Primitive Finitary **S5**-Theory

- Each formula in T is of the form

Muddy Children Example

$$\mathbf{C}(\mathbf{B}_i(m_1 \vee m_2))$$
$$\mathbf{C}(\neg \mathbf{B}_i(m_1 \vee \neg m_2) \wedge \neg \mathbf{B}_i(\neg(m_1 \vee \neg m_2)))$$
$$\mathbf{C}(\neg \mathbf{B}_i(\neg m_1 \vee m_2) \wedge \neg \mathbf{B}_i(\neg(\neg m_1 \vee m_2)))$$
$$\mathbf{C}(\neg \mathbf{B}_i(\neg m_1 \vee \neg m_2) \wedge \neg \mathbf{B}_i(\neg(\neg m_1 \vee \neg m_2)))$$

Properties of Primitive Finitary **S5**-Theories

- Every primitive finitary **S5**-theory T has finitely many finite models such that
  - every model of T is equivalent to one of those finite models of T;
  - for every pair of minimal models $(\mathcal{M}_1, s_1)$ and $(\mathcal{M}_2, s_2)$ of T where with $\mathcal{M}_1 = (M_1, R_1, \pi_1)$ and $\mathcal{M}_2 = (M_2, R_2, \pi_2)$: (i) $|M_1| = |M_2|$; and (ii) for each $u \in M_1$ there exists $v \in M_2$ such that $\pi_1(u) \equiv \pi_2(v)$.
- The set of finite models of T can be computed by eliminating links from the complete graph whose nodes represent the set of worlds satisfying the formulas encoding the true state of the world.

Possible Use

- Epistemic planning problem as a triple (D, I, G) where D is a $mA+$ theory, I is a primitive finitely **S5**-theory, and G is a formula.
- Preliminary implementation of an epistemic planning system. Issues:
  - Data structure for epistemic model, action, states, ...
  - Heuristic

Look back . . .

Research issues in single agent domain

1. Languages for representation of dynamic domains (or reasoning about actions and their effects).
2. Basic algorithms for computing successor states.
3. Search algorithms for plan generation.

Important Notions

1. State
2. Plan

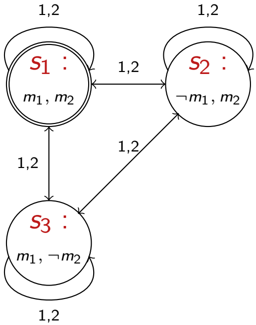
$mA+$ and finitary **S5**-theory provide the basis for extending results from single agent environment to multi-agent environment.

Example

Muddy Children T_1:

$$\mathbf{C}(\mathbf{B}_i(m_1 \vee m_2))$$
$$\mathbf{C}(\neg\mathbf{B}_i(m_1 \vee \neg m_2) \wedge \neg\mathbf{B}_i(\neg(m_1 \vee \neg m_2)))$$
$$\mathbf{C}(\neg\mathbf{B}_i(\neg m_1 \vee m_2) \wedge \neg\mathbf{B}_i(\neg(\neg m_1 \vee m_2)))$$
$$\mathbf{C}(\neg\mathbf{B}_i(\neg m_1 \vee \neg m_2) \wedge \neg\mathbf{B}_i(\neg(\neg m_1 \vee \neg m_2)))$$

Domain: $\{s_1, s_2, s_3\}$



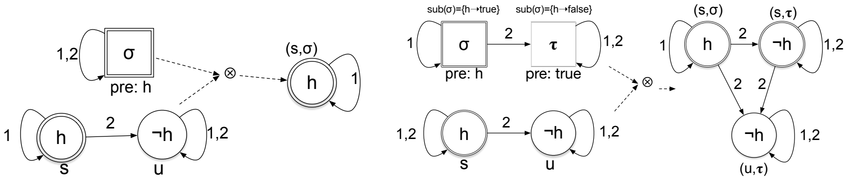
KRR for epistemic planning

From single agent environment to multi-agent environment

- An action language for multi-agent domains for epistemic planning problem specification.
- Suitable restriction yields **KD45** property of epistemic states along a trajectory.

Research issues

- Novel characterization of **KD45** maintainability of $\otimes$ (the current definition of $\otimes$ cannot recover from false beliefs).



- Novel definition of $\otimes$.
- Extension of the action language to consider more complex actions (actions in $mA+$ can be represented by event model with three events).
- Efficient implementation (heuristic, epistemic state equivalent)

Acknowledgements

This presentation contains many results from joint work with Chitta Baral, Michael Gelfond, Enrico Pontelli, and our students (Tu Phan, Son To, Khoi Nguyen, Vien Tran, Gregory Gelfond).

References

Allen, M. and Zilberstein, S. (2009). Complexity of decentralized control: Special cases. In *23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pages 19–27. Curran Associates, Inc.

Aucher, G. and Bolander, T. (2013). Undecidability in epistemic planning. In Rossi, F., editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI.

Baltag, A. and Moss, L. (2004). Logics for epistemic programs. *Synthese*.

Baral, C., Kreinovich, V., and Trejo, R. (2000). Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122:241–267.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840.

Boella, G. and van der Torre, L. W. N. (2005). Enforceable social laws. In Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M. P., and Wooldridge, M., editors, *4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, pages 682–689. ACM.

References (cont.)

Bolander, T., Jensen, M. H., and Schwarzentruber, F. (2015). Complexity results in epistemic planning. In Yang, Q. and Wooldridge, M., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2791–2797. AAAI Press.

Brafman, R. I. and Domshlak, C. (2008). From one to many: Planning for loosely coupled multi-agent systems. In Rintanen, J., Nebel, B., Beck, J. C., and Hansen, E. A., editors, *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, pages 28–35. AAAI.

Brenner, M. (2003). A Multiagent Planning Language. In *Proceedings of Workshop on PDDL, ICAPS*, pages 33–38.

Crosby, M., Jonsson, A., and Rovatsos, M. (2014). A single-agent approach to multiagent planning. In *21st European Conf. on Artificial Intelligence (ECAI'14)*.

de Weerdt, M. and Clement, B. (2009). Introduction to planning in multiagent systems. *Multiagent Grid Systems*, 5:345–355.

Durfee, E. (1999). Distributed Problem Solving and Planning. In *Muliagent Systems (A Modern Approach to Distributed Artificial Intelligence)*, pages 121–164. MIT Press.

Etzioni, O., Golden, K., and Weld, D. (1996). Sound and efficient closed-world reasoning for planning. *Artificial Intelligence*, 89:113–148.

References (cont.)

Fagin, R., Halpern, J., Moses, Y., and Vardi, M. (1995). *Reasoning about Knowledge*. MIT press.

Gelfond, M. and Lifschitz, V. (1993). Representing actions and change by logic programs. *Journal of Logic Programming*, 17(2,3,4):301–323.

Gerbrandy, J. (2006). Logics of propositional control. In Nakashima, H., Wellman, M. P., Weiss, G., and Stone, P., editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 193–200. ACM.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL — the Planning Domain Definition Language. Version 1.2. Technical Report CVC TR98003/DCS TR1165, Yale Center for Comp, Vis and Ctrl.

Goldman, R. and Boddy, M. (1994). Representing uncertainty in simple planners. In *KR 94*, pages 238–245.

Guestrin, C., Koller, D., and Parr, R. (2001). Multiagent planning with factored mdps. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 1523–1530. MIT Press.

Halpern, J., Samet, D., and Segev, E. (2009). Defining knowledge in terms of belief: The modal logic perspective. *Reviews of Symbolic Logic*, 2(3):469–486.

References (cont.)

Herzig, A. and Troquard, N. (2006). Knowing how to play: uniform choices in logics of agency. In Nakashima, H., Wellman, M. P., Weiss, G., and Stone, P., editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 209–216.

Liberatore, P. (1997). The Complexity of the Language *A*. *Electronic Transactions on Artificial Intelligence*, 1(1–3):13–38.

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B. and Michie, D., editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh.

Meyer, J.-J. (2000). Dynamic logic for reasoning about actions and agents. In Minker, J., editor, *Logic-Based Artificial Intelligence*, pages 281–311. Boston/Dordrecht: Kluwer. Chapter 13.

Muise, C., Belle, V., Felli, P., McIlraith, S., Miller, T., Pearce, A. R., and Sonenberg, L. (2015). Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of AAAI*.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D. V., and Marsella, S. (2003). Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In Gottlob, G. and Walsh, T., editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 705–711. Morgan Kaufmann.

References (cont.)

Nissim, R. and Brafman, R. I. (2012). Multi-agent a* for parallel and distributed systems. In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, pages 1265–1266. IFAAMAS.

Peshkin, L. and Savova, V. (2002). Reinforcement learning for adaptive routing. In *Proceedings of the Int. Joint Conf. on Neural Networks*.

Petrick, R. P. A. and Bacchus, F. (2004). Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the Sixth International Conference on Automated Planning and Scheduling, 2004*, pages 2–11.

Poupart, P. and Boutilier, C. (2003). Bounded finite state controllers. In *Neural Information Processing Systems*.

Rathnasabapathy, B., Doshi, P., and Gmytrasiewicz, P. (2006). Exact solutions to interactive POMDPs using behavioral equivalence. In *Autonomous Agents and Multi-Agent Systems Conference (AAMAS)*, pages 1025–1032.

Reiter, R. (2001). *KNOWLEDGE IN ACTION: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press.

Sauro, L., Gerbrandy, J., van der Hoek, W., and Wooldridge, M. (2006). Reasoning about action and cooperation. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 185–192, New York, NY, USA. ACM.

References (cont.)

Shoham, Y. and Leyton-Brown, K. (2011). *MULTIAGENT SYSTEMS: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Pres.

Son, T. C. and Baral, C. (2001). Formalizing sensing actions - a transition function based approach. *Artificial Intelligence*, 125(1-2):19–91.

Son, T. C. and Tu, P. H. (2006). On the Completeness of Approximation Based Reasoning and Planning in Action Theories with Incomplete Information. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning*, pages 481–491.

Son, T. C., Tu, P. H., Gelfond, M., and Morales, R. (2005a). An Approximation of Action Theories of *AL* and its Application to Conformant Planning. In *Proceedings of the the 7th International Conference on Logic Programming and NonMonotonic Reasoning*, pages 172–184.

Son, T. C., Tu, P. H., Gelfond, M., and Morales, R. (2005b). Conformant Planning for Domains with Constraints — A New Approach. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 1211–1216.

Sonu, E. and Doshi, P. (2014). Scalable solutions of interactive pomdps using generalized and bounded policy iteration. *Journal of AAMAS*.

References (cont.)

Spaan, M. T. J., Gordon, G. J., and Vlassis, N. A. (2006). Decentralized planning under uncertainty for teams of communicating agents. In Nakashima, H., Wellman, M. P., Weiss, G., and Stone, P., editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 249–256.

Thielscher, M. (2000). The Fluent Calculus: A Specification Language for Robots with Sensors in Nondeterministic, Concurrent, and Ramifying Environments. Technical Report CL-2000-01, Computational Logic Group, Department of Computer Science, Dresden University of Technology.

To, S. T., Son, T. C., and Pontelli, E. (2015). A generic approach to planning in the presence of incomplete information: Theory and implementation. *Artif. Intell.*, 227:1–51.

Torreño, A., Onaindia, E., and Sapena, O. (2012). An approach to multi-agent planning with incomplete information. In Raedt, L. D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., and Lucas, P. J. F., editors, *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 762–767. IOS Press.

Tran, V., Nguyen, K., Son, T. C., and Pontelli, E. (2013). A conformant planner based on approximation: CpA(H). *ACM TIST*, 4(2):36:1–36:38.

References (cont.)

Tu, P., Son, T., and Baral, C. (2006). Reasoning and planning with sensing actions, incomplete information, and static causal laws using logic programming. *Theory and Practice of Logic Programming*, 7:1–74.

Tu, P., Son, T., Gelfond, M., and Morales, R. (2011). Approximation of action theories and its application to conformant planning. *Artificial Intelligence Journal*, 175(1):79–119.

Turner, H. (2002). Polynomial-length planning spans the polynomial hierarchy. In *Proc. of Eighth European Conf. on Logics in Artificial Intelligence (JELIA'02)*, pages 111–124.

van Benthem, J., van Eijck, J., and Kooi, B. P. (2006). Logics of communication and change. *Inf. Comput.*, 204(11):1620–1662.

van der Hoek, W., Jamroga, W., and Wooldridge, M. (2005). A logic for strategic reasoning. In Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M. P., and Wooldridge, M., editors, *4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, pages 157–164. ACM.

van Ditmarsch, H., van der Hoek, W., and Kooi, B. (2007). *Dynamic Epistemic Logic*. Springer.

Wan, H., Yang, R., Fang, L., Liu, Y., and Xu, H. (2015). A complete epistemic planner without the epistemic closed world assumption. In Yang, Q. and Wooldridge, M., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3257–3263. AAAI Press.